TP Transactions – concurrence d'accès

1 Transactions / v$locked_object / dba_waiters

Il est nécessaire pour ce T.P. de travailler avec plusieurs sessions connectées sur la base de données ora2.
Les manipulations qui suivent attaqueront une table *Donnees(nom varchar2(5), valeur number).* Vous devez donc créer cette table.

Vous initierez cette table de façon à ce qu'elle contienne les tuples: *(A,* 0), *(B,* 0), *(C, 0).*
Par la suite, lorsqu'on écrira:
• « select A » sera une abréviation de « select nom, valeur from donnees where nom=' A' ; »
• « update A(+1) » sera une abréviation de « update donnees set valeur = valeur **+1** where nom ='A' »;
• « select A for update » sera une abréviation de « select * from donnees where nom ='A' for update »;

Ne pas oublier de faire un « commit » sur les ordres de mise à jour !!!

Réaliser les manipulations suivantes, analyser leur comportement. Se référer à la documentation Oracle si nécessaire.

Exécuter '***select distinct sid from v$mystat ;***' pour identifier l'id de votre session

1.1 Les verrous

Exécuter 'select distinct sid from v$mystat ;' pour identifier chacune de vos sessions

Session 1	Session 2	Session 3
select distinct sid from v$mystat ;	select distinct sid from v$mystat ;	select distinct sid from v$mystat ;
Update A(+1)		
	Update A(+1)	
		select * from v$locked_object;'
		select * from dba_waiters ;

SID : Identifiant de la session SQL
SHOW USER permet de récupérer le nom de l'utilisateur de la session SQL

1.2 Etude de ***v$locked_object***

1.2.1 Expliquer les informations récupérées avec la vue ***v$locked_object***

1.3 Etude de la vue **dba_waiters**

1.3.1 Expliquer les informations récupérées avec la vue dba_waiters
1.3.2 La vue dba_waiters permet-elle d'identifier qui attend qui et donc de générer le graphe d'attente ?

2 Transactions Standards

2.1 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
Update A(+1)	
	Select A
commit	
	Select A
	Update A(+1)
Select A	
	commit
	Select A

Quel est ici le phénomène mis en évidence ?

2.2 Manip.

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
Update A(+1)	
	Update B(+1)
	Select A
	Select B
Select A	
Select B	
<t1>	
	Update A(+1)
Update B(+1)	
<t2>	

Commentez les situations aux instants t1 et t2. Identifier la stratégie d'Oracle dans ce type de circonstance.

2.3 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
Select A for update	
<t1>	Update C(+1)
<t2>	Update A(+1)
Update A(+1)	
commit	
Select A	
	commit
	Select A
	Select C

A quoi est due la différence de réaction du système aux instants t1 et t2.

3 Transaction Read Only

3.1 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit*

Session 1	Session 2
Set transaction read only	
<t1> select A	
	Update A(+1)
	commit
<t2> select A	
	Update A(+1)
	Select A
	commit
<t3> select A	
commit	
<t4> select A	

Aux instants *t1, t2 t3, t4* combien de valeurs de *A* au moins sont présentes dans le système, à quoi correspondent-elles?

3.2 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
	Update A(+1)
	Select A
Set transaction read only	
	commit
<t1> select A	
commit	
<t2> select A	

Pourquoi les valeurs de *A* retournées en t1et t2 sont-elles différentes ?

3.3 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
	commit
	Update A(+1)
Set transaction isolation level serializable	
Select A	
Update A(+1)	
Select A	
	Update A(+1)
	commit
Select A	
commit	

Quelle est la différence par rapport à la gestion transactionnelle standard sous Oracle (read commited). Expliquer la fonction du 'Set transaction isolation level serializable'

3.4 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit.*

Session 1	Session 2
	commit
	Update A(+1)
Set transaction isolation level serializable	
Select A	
<t1> Update A(+1)	
	commit
<t2> Update A(+1)	
<t3> Update B(+1)	
Commit	
<t4> Update A(+1)	

Expliquer le comportement du système aux instants t_1, t_2, t_3, t_4.

4 Verrouillage Explicite

4.1 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit*

Session 1	Session 2
Lock table donnees in share mode	
Select A	
	Select A
	Update A(+1)
Select A	
commit	
<t> Select A	

Comment expliquez-vous ce qui s'est passé à l'instant t ? Expliquer le lock share mode par raport au mode read only

4.2 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit*

Session 1	Session 2
Lock table donnees in share mode	
Select A	
	Lock table donnees in share mode
	Select A
Update A(+1)	
	<t> Update B(+1)
Select A	
commit	
Select B	

Comment expliquez-vous ce qui s'est passé à l'instant t ? Expliquer le lock share mode

4.3 Manipulation

Mettre à jour les tuples: *(A,* 0), *(B,* 0), *(C, 0) puis commit*

Session 1	Session 2
Lock table donnees in exclusive mode	
Select A	
	<t1> Select A
	<t2> Select A for update ;
commit	

Comment expliquez-vous ce qui s'est passé aux instants t_1 et t_2? Expliquer le lock exclusive

5 Tableau de compatibilités

Construire le tableau de compatibilités suivantes :

Posé : Demandé :	lockR	lockW1	lockW2
lockR			
lockW1			
lockW2			

Utiliser ‘C’ et ‘I’ :
- C pour Compatible donc pas d’attente entre la session qui pose un verrou et celle qui demande un verrou
- I pour Incompatible donc d’attente entre la session qui pose un verrou et celle qui demande un verrou.

Exécuter les ordres dans trois sessions SQL :
Les ordres SQL suivants seront utilisés :
lockR : lock table donnees in share mode ;
lockW1 : lock table donnees in exclusive mode ;
lockW2 : select * from donnees for update ;

Exécuter ‘select distinct sid from v$mystat ;’ pour identifier chacune de vos sessions
Exécuter dans une 4ème session les vues *v$locked_object* et dba_waiters permettent d’identifier quelles sont les sessions en attente.